Akansha Balhara

- Roll no.: 1/23/SET/BCS/433
- EXP no. : 2
- Date : 29/01/2026

Perceptron learning Algorithm for Implementing AND Logic Gate

```
import numpy as np
# for AND gate (Input/Output)
x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,0,0,1])
epochs = 4
w,b = np.zeros(2),0

for _ in range(epochs):
  for x,y in zip(x_list,y_list):
    z = np.dot(x,w) + b
    ### condition area for y_pred
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    print("error : ",E,"<- at epoch",_)
    w += E*x
    b += E
    print(w,b)
```

```
error :  -1 <- at epoch 0
[0. 0.] -1
error :  0 <- at epoch 0
[0. 0.] -1
error :  0 <- at epoch 0
[0. 0.] -1
error :  1 <- at epoch 0
[1. 1.] 0
error :  -1 <- at epoch 1
[1. 1.] -1
error :  -1 <- at epoch 1
[1. 0.] -2
error :  0 <- at epoch 1
[1. 0.] -2
error :  1 <- at epoch 1
[2. 1.] -1
error :  0 <- at epoch 2
[2. 1.] -1
error :  -1 <- at epoch 2
[2. 0.] -2
error :  -1 <- at epoch 2
[1. 0.] -3
```

```
error :  1 <- at epoch 2
[2. 1.] -2
error :  0 <- at epoch 3
[2. 1.] -2
error :  0 <- at epoch 3
[2. 1.] -2
error :  -1 <- at epoch 3
[1. 1.] -3
error :  1 <- at epoch 3
[2. 2.] -2
```

Change the epochs and observe the output

```
import numpy as np
# for AND gate (Input/Output)
x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,0,0,1])
epochs = 5 # Changed epochs to 10
w,b = np.zeros(2),0

for _ in range(epochs):
  for x,y in zip(x_list,y_list):
    z = np.dot(x,w) + b
    ### condition area for y_pred
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    print("error : ",E,"<- at epoch",_)
    w += E*x
    b += E
    print(w,b)
```

```
error :  -1 <- at epoch 0
[0. 0.] -1
error :  0 <- at epoch 0
[0. 0.] -1
error :  0 <- at epoch 0
[0. 0.] -1
error :  1 <- at epoch 0
[1. 1.] 0
error :  -1 <- at epoch 1
[1. 1.] -1
error :  -1 <- at epoch 1
[1. 0.] -2
error :  0 <- at epoch 1
[1. 0.] -2
error :  1 <- at epoch 1
[2. 1.] -1
error :  0 <- at epoch 2
[2. 1.] -1
```

```
error :  -1 <- at epoch 2
[2. 0.] -2
error :  -1 <- at epoch 2
[1. 0.] -3
error :  1 <- at epoch 2
[2. 1.] -2
error :  0 <- at epoch 3
[2. 1.] -2
error :  0 <- at epoch 3
[2. 1.] -2
error :  -1 <- at epoch 3
[1. 1.] -3
error :  1 <- at epoch 3
[2. 2.] -2
error :  0 <- at epoch 4
[2. 2.] -2
error :  -1 <- at epoch 4
[2. 1.] -3
error :  0 <- at epoch 4
[2. 1.] -3
error :  0 <- at epoch 4
[2. 1.] -3
```

Change the random weights and bias and observe the convergence of the network.

```
import numpy as np
# for AND gate (Input/Output)
x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,0,0,1])
epochs = 7
w,b = [2,1], -1
for _ in range(epochs):
  for x,y in zip(x_list,y_list):
    z = np.dot(x,w) + b
    ### condition area for y_pred
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    print("error : ",E,"<- at epoch",_)
    w += E*x
    b += E
    print(w,b)
```

```
error :  0 <- at epoch 0
[2 1] -1
error :  -1 <- at epoch 0
[2 0] -2
error :  -1 <- at epoch 0
[1 0] -3
error :  1 <- at epoch 0
```

```
[2 1] -2
error :  0 <- at epoch 1
[2 1] -2
error :  0 <- at epoch 1
[2 1] -2
error :  -1 <- at epoch 1
[1 1] -3
error :  1 <- at epoch 1
[2 2] -2
error :  0 <- at epoch 2
[2 2] -2
error :  -1 <- at epoch 2
[2 1] -3
error :  0 <- at epoch 2
[2 1] -3
error :  0 <- at epoch 2
[2 1] -3
error :  0 <- at epoch 3
[2 1] -3
error :  0 <- at epoch 3
[2 1] -3
error :  0 <- at epoch 3
[2 1] -3
error :  0 <- at epoch 3
[2 1] -3
error :  0 <- at epoch 4
[2 1] -3
error :  0 <- at epoch 4
[2 1] -3
error :  0 <- at epoch 4
[2 1] -3
error :  0 <- at epoch 4
[2 1] -3
error :  0 <- at epoch 5
[2 1] -3
error :  0 <- at epoch 5
[2 1] -3
error :  0 <- at epoch 5
[2 1] -3
error :  0 <- at epoch 5
[2 1] -3
error :  0 <- at epoch 6
[2 1] -3
error :  0 <- at epoch 6
[2 1] -3
error :  0 <- at epoch 6
[2 1] -3
error :  0 <- at epoch 6
[2 1] -3
```

Add logic to stop the training when the error is ‘zero’ for all the inputs.

```
import numpy as np
# for AND gate (Input/Output)
x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,0,0,1])
epochs = 10
w,b = [1,2], -1

for _ in range(epochs):
  total_error_in_epoch = 0 # Initialize total error for the current
epoch
  for x,y in zip(x_list,y_list):
    z = np.dot(x,w) + b
    ### condition area for y_pred
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    total_error_in_epoch += abs(E) # Accumulate absolute error
    print("error : ",E,"<- at epoch",_)
    w += E*x
    b += E
    print(w,b)
  if total_error_in_epoch == 0:
    print(f"Converged at epoch {_}. No errors found.")
    break # Stop training if no errors occurred in the epoch
```

```
error :  0 <- at epoch 0
[1 2] -1
error :  -1 <- at epoch 0
[1 1] -2
error :  0 <- at epoch 0
[1 1] -2
error :  0 <- at epoch 0
[1 1] -2
error :  0 <- at epoch 1
[1 1] -2
error :  0 <- at epoch 1
[1 1] -2
error :  0 <- at epoch 1
[1 1] -2
error :  0 <- at epoch 1
[1 1] -2
Converged at epoch 1. No errors found.
```

Repeat the same process for OR gate

```
import numpy as np
# for OR gate (Input/Output)
x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,1,1,1]) # Changed y_list for OR gate
```

```python
epochs = 10
w,b = [1,2], 2

for _ in range(epochs):
  total_error_in_epoch = 0
  for x,y in zip(x_list,y_list):
    z = np.dot(x,w) + b
    ### condition area for y_pred
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    total_error_in_epoch += abs(E)
    print("error : ",E,"<- at epoch",_)
    w += E*x
    b += E
    print(w,b)
  if total_error_in_epoch == 0:
    print(f"Converged at epoch {_}. No errors found.")
    break
```

```
error :  -1 <- at epoch 0
[1 2] 1
error :  0 <- at epoch 0
[1 2] 1
error :  0 <- at epoch 0
[1 2] 1
error :  0 <- at epoch 0
[1 2] 1
error :  -1 <- at epoch 1
[1 2] 0
error :  0 <- at epoch 1
[1 2] 0
error :  0 <- at epoch 1
[1 2] 0
error :  0 <- at epoch 1
[1 2] 0
error :  -1 <- at epoch 2
[1 2] -1
error :  0 <- at epoch 2
[1 2] -1
error :  0 <- at epoch 2
[1 2] -1
error :  0 <- at epoch 2
[1 2] -1
error :  0 <- at epoch 3
[1 2] -1
error :  0 <- at epoch 3
[1 2] -1
error :  0 <- at epoch 3
[1 2] -1
error :  0 <- at epoch 3
```

```
[1 2] -1
Converged at epoch 3. No errors found.
```

Try the process for XOR gate and show that the network will not stabilize (Converge).

```python
import numpy as np
# for XOR gate (Input/Output)
x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,1,1,0]) # Changed y_list for XOR gate
epochs = 15 # Increased epochs to clearly show non-convergence
w,b = [1,2], 2

for _ in range(epochs):
  total_error_in_epoch = 0
  for x,y in zip(x_list,y_list):
    z = np.dot(x,w) + b
    ### condition area for y_pred
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    total_error_in_epoch += abs(E)
    print("error : ",E,"<- at epoch",_)
    w += E*x
    b += E
    print(w,b)
  if total_error_in_epoch == 0:
    print(f"Converged at epoch {_}. No errors found.")
    break
```

```
error :  -1 <- at epoch 0
[1 2] 1
error :  0 <- at epoch 0
[1 2] 1
error :  0 <- at epoch 0
[1 2] 1
error :  -1 <- at epoch 0
[0 1] 0
error :  -1 <- at epoch 1
[0 1] -1
error :  0 <- at epoch 1
[0 1] -1
error :  1 <- at epoch 1
[1 1] 0
error :  -1 <- at epoch 1
[0 0] -1
error :  0 <- at epoch 2
[0 0] -1
error :  1 <- at epoch 2
[0 1] 0
error :  0 <- at epoch 2
```

```
[0 1] 0
error :  -1 <- at epoch 2
[-1  0] -1
error :  0 <- at epoch 3
[-1  0] -1
error :  1 <- at epoch 3
[-1  1] 0
error :  1 <- at epoch 3
[0 1] 1
error :  -1 <- at epoch 3
[-1  0] 0
error :  -1 <- at epoch 4
[-1  0] -1
error :  1 <- at epoch 4
[-1  1] 0
error :  1 <- at epoch 4
[0 1] 1
error :  -1 <- at epoch 4
[-1  0] 0
error :  -1 <- at epoch 5
[-1  0] -1
error :  1 <- at epoch 5
[-1  1] 0
error :  1 <- at epoch 5
[0 1] 1
error :  -1 <- at epoch 5
[-1  0] 0
error :  -1 <- at epoch 6
[-1  0] -1
error :  1 <- at epoch 6
[-1  1] 0
error :  1 <- at epoch 6
[0 1] 1
error :  -1 <- at epoch 6
[-1  0] 0
error :  -1 <- at epoch 7
[-1  0] -1
error :  1 <- at epoch 7
[-1  1] 0
error :  1 <- at epoch 7
[0 1] 1
error :  -1 <- at epoch 7
[-1  0] 0
error :  -1 <- at epoch 8
[-1  0] -1
error :  1 <- at epoch 8
[-1  1] 0
error :  1 <- at epoch 8
[0 1] 1
```

```
error :  -1 <- at epoch 8
[-1  0] 0
error :  -1 <- at epoch 9
[-1  0] -1
error :  1 <- at epoch 9
[-1  1] 0
error :  1 <- at epoch 9
[0 1] 1
error :  -1 <- at epoch 9
[-1  0] 0
error :  -1 <- at epoch 10
[-1  0] -1
error :  1 <- at epoch 10
[-1  1] 0
error :  1 <- at epoch 10
[0 1] 1
error :  -1 <- at epoch 10
[-1  0] 0
error :  -1 <- at epoch 11
[-1  0] -1
error :  1 <- at epoch 11
[-1  1] 0
error :  1 <- at epoch 11
[0 1] 1
error :  -1 <- at epoch 11
[-1  0] 0
error :  -1 <- at epoch 12
[-1  0] -1
error :  1 <- at epoch 12
[-1  1] 0
error :  1 <- at epoch 12
[0 1] 1
error :  -1 <- at epoch 12
[-1  0] 0
error :  -1 <- at epoch 13
[-1  0] -1
error :  1 <- at epoch 13
[-1  1] 0
error :  1 <- at epoch 13
[0 1] 1
error :  -1 <- at epoch 13
[-1  0] 0
error :  -1 <- at epoch 14
[-1  0] -1
error :  1 <- at epoch 14
[-1  1] 0
error :  1 <- at epoch 14
[0 1] 1
```

```
error :  -1 <- at epoch 14
[-1  0] 0
```